On the complexity of real root isolation using Continued Fractions

Elias P. Tsigaridas, Ioannis Emiris

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On the complexity of real root isolation using Continued Fractions

Elias P. Tsigaridas — Ioannis Z. Emiris

N° 6059

Nobember 2006

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6059--FR+ENG

On the complexity of real root isolation using Continued Fractions

Elias P. Tsigaridas* , Ioannis Z. Emiris†

Thème SYM — Systèmes symboliques
Projet GEOMETRICA

Rapport de recherche n° 6059 — Nobember 2006 — 22 pages

Abstract: We present algorithmic, complexity and implementation results concerning real root isolation of integer univariate polynomials using the continued fraction expansion of real algebraic numbers. One motivation is to explain the method's good performance in practice. We improve the previously known bound by a factor of $d\tau$, where d is the polynomial degree and τ bounds the coefficient bit size, thus matching the current record complexity for real root isolation by exact methods (Sturm, Descartes and Bernstein subdivision). Namely our complexity bound is $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ using a standard bound on the expected bit size of the integers in the continued fraction expansion. Moreover, using a homothetic transformation we improve the expected complexity bound to $\widetilde{\mathcal{O}}_B(d^3\tau)$ under the assumption that $d = \mathcal{O}(\tau)$. We compute the multiplicities within the same complexity and extend the algorithm to non square-free polynomials. Finally, we present an efficient open-source `C++` implementation and illustrate its completeness and efficiency as compared to other available software. For this we use polynomials with coefficient bit size up to 8000 bits and degree up to 1000.

Key-words: continued fractions, real root isolation, Descartes' rule of sign, bit complexity, separation bound

This work is partially supported by the european project ACS (Algorithms for Complex Shapes, IST FET Open 006413) and ARC ARCADIA (http://www.loria.fr/∼petitjea/Arcadia/)

* Part of this work was done while the author was a PhD student at the Department of Informatics and Telecommunications of National Kapodistrian University of Athens, HELLAS

† Department of Informatics and Telecommunications National Kapodistrian University of Athens, HELLAS

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

On the complexity of real root isolation using Continued Fractions

Résumé : Pas de résumé

Mots-clés : Pas de motclef

1 Introduction

In this paper we deal with real root isolation of univariate integer polynomials, a fundamental problem in computer algebra as well as in many applications ranging from computational geometry to quantifier elimination. The problem consists in computing intervals with rational endpoints which contain exactly one real root of the polynomial and have such an interval for every real root. We use the continued fraction expansion of *real algebraic numbers*. Recall that such a number is a real root of an integer polynomial.

A major motivation is to explain the method's good performance in implementations, despite the higher complexity bounds which were known until now. Indeed, we show that continued fractions lead to asymptotic bit complexity bounds that match those recently proven for other exact methods, such as Sturm, Descartes and Bernstein subdivision algorithms. Using results from the metric theory of continued fractions we prove that the algorithm achieves an expected complexity of $\widetilde{\mathcal{O}}_B(d^4\tau^2)$, where d is the polynomial degree and τ bounds the coefficient bit size, thus we improve the previous known bound by a factor of $d\tau$. Moreover, under the hypothesis that $d = \mathcal{O}(\tau)$, we present a variant of the algorithm with complexity $\widetilde{\mathcal{O}}_B(d^3\tau)$.

1.1 Notation

In what follows $\mathcal{O}_B$ means bit complexity and the $\widetilde{\mathcal{O}}_B$-notation means that we are ignoring logarithmic factors. For a polynomial $A = \sum_{i=1}^{d} a_i X^i \in \mathbb{Z}[X]$, $\mathtt{deg}(A)$ denotes its degree. We consider square-free polynomials except if explicitly stated otherwise. By $\mathcal{L}(A)$ we denote an upper bound on the bit size of the coefficients of A (including a bit for the sign). For $\mathsf{a} \in \mathbb{Q}$, $\mathcal{L}(\mathsf{a}) \geq 1$ is the maximum bit size of the numerator and the denominator. Let $\mathsf{M}(\tau)$ denote the bit complexity of multiplying two integers of bit size at most τ. Using FFT, $\mathsf{M}(\tau) = \mathcal{O}_B(\tau \lg^c \tau)$ for a suitable constant c. $Var(A)$ denotes the number of sign variations in the coefficient list of A ignoring zero terms and Δ the separation bound of A, that is the smallest distance between two (complex) roots of A.

1.2 Previous work and our results

Real root isolation of univariate integer polynomials is a well known problem and various algorithms exist for it. Moreover, there is a huge bibliography on the problem so we have to mention that we only scratch the surface of the existing literature and we encourage the reader to refer to the references.

Most exact subdivision based algorithms for real root isolation are based either on Descartes' rule of sign (Th. 1) or on Sturm sequences. Roughly speaking, the idea behind both approaches is to subdivide a given interval that initially contains all the real roots until it is certified that none or one real root is contained in the tested interval. Descartes' approach achieves this by repeatedly transforming the original polynomial and counting the sign variations in the coefficients' list, while Sturm's approach computes a signed polynomial

remainder sequence and evaluates it over the endpoints of the interval of interest. Quite recently it was proven (c.f [18, 19, 21] and references therein) that both approaches, the one based on Descartes' rule of sign (where the polynomials are represented either in the monomial or in the Bernstein basis) and the one based on Sturm sequences, achieve the same bit complexity bound, namely $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ or $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$. Moreover, using Sturm(-Habicht) sequences in a pre-processing and a post-processing step [21] the bound holds for the non square-free case and the multiplicities of the roots can also be computed. If the degree of the polynomial is ≤ 4 then real solving can be performed in $\mathcal{O}(1)$ or $\widetilde{\mathcal{O}}_B(\tau)$ [20].

The continued fraction algorithm (from now on called CF) differs from the subdivision based algorithms in that instead of bisecting a given initial interval it computes the continued fraction expansion for each real root of the polynomial. The first formulation of the algorithm is due to Vincent [47], see also [1, 6] for historical references. It was based on his theorem (Th. 4 without the terminating condition) where it was stated that repeated transformations of the polynomial will eventually yield a polynomial with zero (or one) sign variation, thus Descartes' rule (Th. 1 and Rem. 2) implies that the transformed polynomial has zero (resp. one) real root in $(0, \infty)$. If one sign variation is attained then the inverse transformation can be applied in order to compute an isolating interval for the real root that corresponds to the original polynomial. Moreover, the integers, c_i's, used in the transformations correspond to the partial quotients of the continued fraction expansion of the real root. However, Vincent's algorithm is exponential [15]. He computed the c_i's in the transformation of Th. 4 by repeated shift operations of the form $X \mapsto X + 1$, thus if one of the c_i's (or even the sum of all) is of magnitude, say, 2^τ then an exponential number of steps must be performed.

Uspensky [44] extended Vincent's theorem by computing an upper bound on the number of transformations so as to isolate the real roots, but failed to deal with its exponential behavior. See also [13, 39] where the problem of approximating a real algebraic number is also considered. Using Vincent's theorem, Collins and Akritas [15] derived a polynomial subdivision-based algorithm using Descartes' rule of sign.

Akritas [2, 5] dealt with the exponential behavior of the CF algorithm, by computing the c_i's in the transformations as positive lower bounds of the positive real roots, via Cauchy's bound (for details, see sec. 3). He achieved a complexity of $\widetilde{\mathcal{O}}_B(d^5\tau^3)$ or $\widetilde{\mathcal{O}}_B(N^8)$, without using fast Taylor shifts [48]. However, it is not clear how this approach accounts for the increased coefficient size in the transformed polynomial after applying a map of the form $X \mapsto b + X$. Another issue is to bound the size of the c_i. Refer to Eq. (1) which indicates that the *magnitude* of the partial quotients is unbounded. CF is the standard real root isolation function in MATHEMATICA [4] and for some experiments against subdivision-based algorithms, also in MATHEMATICA, the reader may refer to [3].

Another class of univariate solvers are numerical solvers, e.g [36, 37, 41], that compute an approximation of all the roots (real and complex) of a polynomial up to a desired accuracy. The complexity of these algorithms is $\widetilde{\mathcal{O}}_B(d^3\,\tau)$.

The contributions of this paper are the following: First, we improve the bound of the number of steps (transformations) that the CF algorithm performs. The proof of this is

achieved through Th. 6. Second, we bound the bit size of the partial quotients and thus the growth of the transformed polynomials which appear during the algorithm. For this we use the hypothesis of the continued fraction expansion of real numbers and a standard average case analysis. We revisit the proof of [2, 5] so as to improve the overall bit complexity bound of the algorithm to $\widetilde{\mathcal{O}}_B(N^6)$, thus matching the current record complexity for exact real root isolation. From a theoretical perspective, we present a variant of the CF algorithm which, under the hypothesis that $d = \mathcal{O}(\tau)$, has expected complexity $\widetilde{\mathcal{O}}_B(N^4)$, thus matching the complexity of the numerical algorithms. The extension to the non square-free case uses the techniques from [21]. Finally, we present our efficient open-source `C++` implementation of the $\widetilde{\mathcal{O}}_B(N^6)$ algorithm and illustrate it on various data sets, including polynomials of degree up to 1000 and coefficients of 8000 bits. Our software seems comparable to, and some times faster than the root isolation implementations that we tested, including RS[1], which seems to be one of the fastest available software for exact real root isolation. We also tested a numeric solver, namely ABERTH [9, 10], which is very efficient in practice but needs special tuning in order to produce the correct number of real roots. We believe that our software contributes towards reducing the gap between rational and numeric computation, the latter being usually perceived as faster.

Part of this work appeared in [43].

The rest of the paper is structured as follows. The next section sketches the theory behind continued fractions. Sec. 3 presents the CF algorithm and Sec. 4 its analysis. We conclude with experiments using our implementation, along with comparisons against other available software for univariate equation solving.

2 Continued fractions

We present a short introduction to continued fractions, following [45] which, although is far from complete, suffices for our purposes. The reader may refer to e.g [5, 11, 45, 50]. In general a *simple (regular) continued fraction* is a (possibly infinite) expression of the form

$$c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \dots}} = [c_0, c_1, c_2, \dots],$$

where the numbers c_i are called *partial quotients*, $c_i \in \mathbb{Z}$ and $c_i \geq 1$ for $i > 0$. Notice that c_0 may have any sign, however in our real root isolation algorithm $c_0 \geq 0$, without loss of generality. By considering the recurrent relations

$$\begin{array}{lll} P_{-1} = 1, & P_0 = c_0, & P_{n+1} = c_{n+1}\, P_n + P_{n-1}, \\ Q_{-1} = 0, & Q_0 = 1, & Q_{n+1} = c_{n+1}\, Q_n + Q_{n-1}, \end{array}$$

[1] `http://fgbrs.lip6.fr/salsa/Software/`

it can be shown by induction that $R_n = \frac{P_n}{Q_n} = [c_0, c_1, \ldots, c_n]$, for $n = 0, 1, 2, \ldots$ and moreover that

$$\begin{aligned} P_n\, Q_{n+1} - P_{n+1}\, Q_n &= (-1)^{n+1}, \\ P_n\, Q_{n+2} - P_{n+2}\, Q_n &= (-1)^{n+1} c_{n+2}. \end{aligned}$$

If $\gamma = [c_0, c_1, \ldots]$ then $\gamma = c_0 + \frac{1}{Q_0 Q_1} - \frac{1}{Q_1 Q_2} + \cdots = c_0 + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{Q_{n-1} Q_n}$ and since this is a series of decreasing alternating terms it converges to some real number γ. A finite section $R_n = \frac{P_n}{Q_n} = [c_0, c_1, \ldots, c_n]$ is called the $n-$th *convergent* (or *approximant*) of γ and the tails $\gamma_{n+1} = [c_{n+1}, c_{n+2}, \ldots]$ are known as its *complete quotients*. That is $\gamma = [c_0, c_1, \ldots, c_n, \gamma_{n+1}]$ for $n = 0, 1, 2, \ldots$. There is a one to one correspondence between the real numbers and the continued fractions, where evidently the finite continued fractions correspond to rational numbers.

It is known that $Q_n \geq F_{n+1}$ and that $F_{n+1} < \phi^n < F_{n+2}$, where F_n is the $n-$th Fibonacci number and $\phi = \frac{1+\sqrt{5}}{2}$ is the *golden ratio*. Continued fractions are the best (for a given denominator size), approximation. This is as follows:

$$\frac{1}{Q_n(Q_{n+1} + Q_n)} \leq \left| \gamma - \frac{P_n}{Q_n} \right| \leq \frac{1}{Q_n Q_{n+1}} < \phi^{-2n+1}.$$

Let $\gamma = [c_0, c_1, \ldots]$ be the continued fraction expansion of a real number. The Gauss-Kuzmin distribution [11, 38] states that for almost all real numbers γ (meaning that the set of exceptions has Lebesgue measure zero) the probability for a positive integer δ to appear as an element c_i in the continued fraction expansion of γ is

$$Prob[c_i = \delta] = \lg \frac{(\delta+1)^2}{\delta(\delta+2)}, \quad \text{for any fixed } i > 0. \tag{1}$$

The Gauss-Kuzmin law induces that we can not bound the mean value of the partial quotients or in other words that the expected value (arithmetic mean) of the partial quotients is diverging, i.e

$$E[c_i] = \sum_{\delta=1}^{\infty} \delta\, Prob[c_i = \delta] = \infty, \text{ for } i > 0.$$

Surprisingly enough the geometric (and the harmonic) mean is not only asymptotically bounded, but is bounded by a constant, for almost all $\gamma \in \mathbb{R}$. For the geometric mean this is the famous Khintchine's constant [27], i.e.

$$\lim_{n\to\infty} \sqrt[n]{\prod_{i=1}^{n} c_i} = \mathcal{K} = 2.685452001...$$

which is not known if it is an irrational number, let alone transcendental. The reader may refer to [7] for a comprehensive treatment of *Khintchine's means*. The expected value of the

bit size of the partial quotients is a constant for almost all real numbers, when $n \to \infty$ or n sufficiently big [27, 38]. Following closely [38], we have:

$$E[\ln c_i] = \frac{1}{n}\sum_{i=1}^{n} \ln c_i = \ln \mathcal{K} = 0.98785..., \text{ as } n \to \infty, \forall\, i > 0.$$

Let $\mathcal{L}(c_i) \triangleq b_i$, then

$$E[b_i] = \mathcal{O}(1). \tag{2}$$

A real number has an (eventually) periodic continued fraction expansion if and only if it is a root of an irreducible quadratic polynomial. The set of real algebraic numbers is countable and has Lebesgue measure zero, thus there is chance that Gauss-Kuzmin distribution and Khintchine's law do not hold for it. However, "There is no reason to believe that the continued fraction expansions of non-quadratic algebraic irrationals generally do anything other than faithfully follow Khintchine's law"[12]. Moreover, various experimental results [11, 38, 39] support the conjecture. It is a major open problem to find an irreducible integer polynomial such that the continued fraction expansions of its real roots do not follow the conjecture or to prove the conjecture.

For the largest digit that can appear in the partial quotients of a rational number the reader may refer to [24].

For our analysis we rely on the *conjecture* that Gauss-Kuzmin's distribution and Khintchine's law hold for the set of real algebraic numbers. If it can proven that the partial quotients of the continued fraction expansion of real algebraic numbers are bounded this will lead to an improvement of our complexity bounds.

3 The CF algorithm

Theorem 1 (Descartes' rule of sign) *The number R of real roots of $A(X)$ in $(0, \infty)$ is bounded by $Var(A)$ and we have $R \equiv Var(A) \mod 2$.*

Remark 2 *In general Descartes' rule of sign obtains an overestimation of the number of the positive real roots. However, if we know that A is* hyperbolic*, i.e has only real roots or when the number of sign variations is 0 or 1 then it counts exactly.*

The proof of Th. 1 follows from the following theorem which is due to Budan:

Theorem 3 (Budan) [5, 32] *Let a polynomial A, such that $\deg(A) = d$ and let $a < b$, where $a, b \in \mathbb{R}$. Let A_a, resp. A_b, be the polynomial produced after we apply the map $X \mapsto X + a$, resp. $X \mapsto X + b$, to A. Then the following hold:*

1. $Var(A_a) \geq Var(A_b)$,
2. $\#\{\gamma \in (a, b) | A(\gamma) = 0\} \leq Var(A_a) - Var(A_b)$, *and*
3. $\#\{\gamma \in (a, b) | A(\gamma) = 0\} \equiv Var(A_a) - Var(A_b) \mod 2$.

Algorithm 1: $\texttt{CF}(A, M)$

Input: $A \in \mathbb{Z}[X], M(X) = \frac{kX+l}{mX+n}$, $k, l, m, n \in \mathbb{Z}$
Output: A list of isolating intervals

```
1  if A(0) = 0 then
2  |  OUTPUT Interval( M(0), M(0)) ;
3  |  A ← A(X)/X;
4  |  CF(A, M);
5  V ← Var(A);
6  if V = 0 then RETURN ;
7  if V = 1 then
8  |  OUTPUT Interval( M(0), M(∞));
9  |  RETURN ;
10 b ← PLB(A) // PLB ≡ PositiveLowerBound ;
11 if b > 1 then  A ← A(b + X), M ← M(b + X) ;
12 A1 ← A(1 + X), M1 ← M(1 + X) ;
13 CF(A1, M1) // Looking for real roots in (1, +∞);
14 A2 ← A(1/(1+X)), M2 ← M(1/(1+X)) ;
15 CF(A2, M2) // Looking for real roots in (0, 1) ;
16 RETURN ;
```

The CF algorithm depends on the following theorem, which dates back to Vincent's theorem in 1836 [47]. The inverse of Th. 4 can be found in [5, 16, 32]. It is a very interesting question whether the one and two circle theorems (c.f [31] and references therein), employed in the analysis of the Descartes/Bernstein algorithm [15], can also be applied and possibly improve the complexity of the CF algorithm. The version of the theorem that we present is due to Alesina and Galuzzi [6] and improves the conditions of all the previous versions [1, 2, 5, 44].

Theorem 4 [6] *Let $A \in \mathbb{Z}[X]$ be square-free and let $\Delta > 0$ be the separation bound. Let n be the smallest index such that*

$$F_{n-1}\, F_n\, \Delta > \frac{2}{\sqrt{3}},$$

where F_n is the n-th Fibonnaci number. Then the map $X \mapsto [c_0, c_1, \ldots, c_n, X]$, where $c_0, c_1, \ldots, c_n$ is an arbitrary sequence of positive integers, transforms $A(X)$ to $A_n(X)$, whose list of coefficients has no more than one sign variation.

Notice that in the previous theorem the conditions do not depend on the degree of the polynomial. Moreover, a similar theorem holds for non square-free polynomials but we will not use it for the analysis of the CF algorithm. The extension of Vincent's theorem to the

non square-free case is due to Wang [49], see also [14] and for an improved version and historical references see [6].

Theorem 5 [6, 49] *Let $A \in \mathbb{Z}[X]$, not necessarily square-free, with $\mathtt{deg}(A) = d$ and let $\Delta > 0$ be the separation bound. Let k be the smallest index such that $F_{k-1}^2 \Delta > 1$, m be the smallest integer such that $m > \frac{1}{2}\log_\phi d$ and $n = k + m$.*

Then the map $X \mapsto [c_0, c_1, \ldots, c_n, X]$, where $c_0, c_1, \ldots, c_n$ is an arbitrary sequence of positive integers, transforms $A(X)$ to $A_n(X)$, that has $Var(A_n) \geq 0$ sign variations. If $Var(A_n) > 0$ then A_n has a unique positive real root of multiplicity $Var(A_n)$.

The previous extension of Vincent's theorem implies that Descartes' rule of sign can be used to isolate the real roots of non square-free polynomials and to compute their multiplicities, contrary to what it is believed up to now. Moreover, Th. 5 implies that Descartes' rule of sign can be used for polynomials with multiprocessing floating point and/or interval coefficients. We plan to report on the consequences of Th. 5 in a future work. Of course the obstacle to all the previous remarks is that we have to perform iterations up to the theoretical separation bound, which is a very bad overestimation of the actual one.

In our analysis we will assume that the input polynomial is square-free, except if explicitly stated otherwise, since we compute the multiplicities of the real roots differently. Thus we will rely on Th. 4 to isolate the positive real roots of a square-free polynomial A. In order to isolate the negative roots we perform the transformation $X \mapsto -X$, so in what follows we will consider only the positive real roots of A.

Vincent's variant of the CF algorithm goes as follows: A polynomial A is transformed to A_1 by the transformation $X \mapsto 1 + X$ and if $Var(A_1) = 0$ or $Var(A_1) = 1$ then A has 0, resp. 1, real root greater than 1 (Th. 1). If $Var(A_1) < Var(A)$ then (possibly) there are real roots of A in $(0, 1)$, due to Budan's theorem (Th. 3). A_2 is produced by applying the transformation $X \mapsto 1/(1 + X)$ to A. If $Var(A_2) = 0$ or $Var(A_2) = 1$ then A has 0, resp. 1, real root less than 1 (Th. 1). Uspensky's [44] variant of the algorithm (see also [39]) at every step produces both polynomials A_1 and A_2 probably, as Akritas states [1], because he was unaware of Budan's theorem. In both variants, if the transformed polynomial has more than one sign variations, we repeat the process.

We may consider the process of the algorithm as an infinite binary tree in which the root corresponds to the original polynomial A. The branch from a node to a right child corresponds to the map $X \mapsto X + 1$, while to the left child to the map $X \mapsto \frac{1}{1+X}$. Notice that a sequence of c transformations $X \mapsto 1+X$ followed by one of the type $X \mapsto 1/(1+X)$ is equivalent to two transformations, one of the type $X \mapsto c+1/X$ followed by $X \mapsto 1+X$. Thus Vincent's algorithm (and Uspensky's) results to a sequence of transformations like the one described in Th. 4, and so the leaves of the binary tree that we considered hold (transformed) polynomials that have no more than one sign variations, if Th. 4 holds. Akritas [2, 5] replaced a series of $X \mapsto X + 1$ transformations by $X \mapsto X + b$, where b is the positive lower bound (`PLB`) on the positive roots of the tested polynomial. This was computed by Cauchy's bound [5, 32, 50]. This way, the number of steps is polynomial and the complexity is in $\widetilde{\mathcal{O}}_B(d^5\tau^3)$.

However, it is not clear whether or how the analysis takes into account that the coefficient bit size increases after a shift. Another issue is to bound the size of the b's.

For these polynomials that have one sign variation we still have to find the interval where the real root of the initial polynomial A lies. Consider a polynomial A_n that corresponds to a leaf of the binary tree that has one sign variation. Notice that A_n is produced after a transformation as in Th. 4, using positive integers $c_0, c_1, \ldots, c_n$. This transformation can be written in a more compact form using the convergents

$$M : X \mapsto \frac{P_n X + P_{n-1}}{Q_n X + Q_{n-1}}, \tag{3}$$

where $\frac{P_{n-1}}{Q_{n-1}}$ and $\frac{P_n}{Q_n}$ are consecutive convergents of the continued fraction $[c_0, c_1, \ldots, c_n]$. Notice that (3) is a Möbius transformation, see [5, 50] for more details. Since A_n has one sign variation it has one and only one real root in $(0, \infty)$, so in order to obtain the isolating interval for the corresponding real root of A we evaluate the right part of Eq. (3) once over 0 and once over ∞. The (unordered) endpoints of the isolating interval are $\frac{P_{n-1}}{Q_{n-1}}$ and $\frac{P_n}{Q_n}$.

The pseudo-code of the CF algorithm is presented in Alg. 1. Notice that the `Interval` function orders the endpoints of the computed isolating interval and that $\texttt{PLB}(A)$ computes a lower bound on the positive roots of A. The initial input of the algorithm is a polynomial $A(X)$ and the trivial transformation $M(X) = X$. We need the functional M in order to keep track of the transformations that we perform so that to derive the isolating intervals. Notice that Line 15 is to be executed only when $Var(A_1) < Var(A_2)$, but in order to simplify the analysis we omit this, since it only doubles the complexity.

4 The complexity of the CF algorithm

The complexity of the CF algorithm depends on the number of transformations and the cost of each. However, special care should be taken since after each transformation the bit size of the polynomial coefficients increases.

Let $\texttt{disc}(A)$ be the discriminant and $\texttt{lead}(A)$ the leading coefficient of A. Mahler's measure of a polynomial A is $\mathcal{M}(A) = |\,\texttt{lead}\,(A)\,| \prod_{i=1}^{d} \max\{1, |\gamma_i|\}$, where γ_i are all the (complex) roots of A [8, 32, 33, 50]. Moreover $\mathcal{M}(A) \leq 2^{\tau}\sqrt{d+1}$. We prove the following theorem, which is based on a theorem by Mignotte [32], thus extending [17, 19].

Theorem 6 *Let $A \in \mathbb{Z}[X]$, with $\deg(A) = d$ and $\mathcal{L}(A) = \tau$. Let Ω be any set of k couples of indices (i, j) such that $1 \leq i < j \leq d$ and let the non-zero (complex) roots of A be $0 < |\gamma_1| \leq |\gamma_2| \leq \cdots \leq |\gamma_d|$. Then*

$$2^k \mathcal{M}(A)^k \geq \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \geq 2^{k - \frac{d(d-1)}{2}}\, \mathcal{M}(A)^{1-d-k}\, \sqrt{\texttt{disc}(A)}$$

Proof. Consider the multiset $\overline{\Omega} = \{j | (i, j) \in \Omega\}$, where $|\overline{\Omega}| = k$. We use the inequality

$$\forall\, a, b \in \mathbb{C} \quad |a - b| \leq 2 \max\{|a|, |b|\}, \tag{4}$$

and the fact [32, 33] that for any root of A, $\frac{1}{\mathcal{M}(A)} \leq |\gamma_i| \leq \mathcal{M}(A)$. In order to prove the left inequality

$$\prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \leq 2^k \prod_{j\in\overline{\Omega}} |\gamma_j| \leq 2^k \max_{j\in\overline{\Omega}} |\gamma_j|^k \leq 2^k \mathcal{M}(A)^k.$$

Recall [32, 50] that $\mathtt{disc}(A) = \mathtt{lead}\,(A)^{2d-2} \prod_{i<j} (\gamma_i - \gamma_j)^2$. For the right inequality we consider the absolute value of the discriminant of A, i.e

$$\begin{array}{rcl} |\,\mathtt{disc}(A)| & = & |\,\mathtt{lead}\,(A)\,|^{2d-2} \prod_{i<j} |\gamma_i - \gamma_j|^2 \\ & = & |\,\mathtt{lead}\,(A)\,|^{2d-2} \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j|^2 \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j|^2 \quad \Leftrightarrow \\ \sqrt{|\,\mathtt{disc}(A)|} & = & |\,\mathtt{lead}\,(A)\,|^{d-1} \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j| \end{array}$$

We consider the product $\prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j|$ and we apply $\frac{d(d-1)}{2} - k$ times inequality (4), thus

$$\begin{array}{rcl} \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j| & \leq & 2^{\frac{d(d-1)}{2}-k} |\gamma_1|^0 |\gamma_2|^1 \cdots |\gamma_d|^{d-1} \left(\prod_{j\in\overline{\Omega}} |\gamma_j|\right)^{-1} \\ & \leq & 2^{\frac{d(d-1)}{2}-k} \mathcal{M}(A)^{d-1} |\,\mathtt{lead}\,(A)\,|^{1-d} \mathcal{M}(A)^k \end{array} \tag{5}$$

where we used the inequality $|\gamma_1|^0 |\gamma_2|^1 \cdots |\gamma_d|^{d-1} \leq |\mathcal{M}(A)/\,\mathtt{lead}\,(A)\,|^{d-1}$, and the fact [32] that, since $\forall i$, $|\gamma_i| \geq \mathcal{M}(A)^{-1}$, we have $\prod_{j\in\overline{\Omega}} |\gamma_j| \geq |\gamma_1|^k \geq \mathcal{M}(A)^{-k}$. Thus we conclude that

$$\prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \geq 2^{k-\frac{d(d-1)}{2}} \mathcal{M}(A)^{1-d-k} \sqrt{|\,\mathtt{disc}(A)|}.$$

□

A similar theorem but with more strict hypotheses on the roots first appeared in [17], see also [26], and the conditions were generalized in [19]; namely in order for the bound [17, 19] to hold the sets of indices i and j should be rearranged such that they form an acyclic graph where each node has out-degree at most one. The bound of Th. 6 has a factor 2^{d^2} instead of d^d in [17, 19, 26], which plays no role when $d = \mathcal{O}(\tau)$ or when the notation with N is used. Moreover, we loosen the hypotheses of the theorem and thus all the proofs concerning the number of steps of the subdivision-based solvers [19, 21] are dramatically simplified. Possibly a more involved proof of Th. 6 may eliminate this factor using [34].

Remark 7 *There are two simple, however crucial, observations about Th. 4. When the transformed polynomial has one sign variation, then the interval with endpoints* $\frac{P_{n-1}}{Q_{n-1}} = [c_0, c_1, \ldots, c_{n-1}]$ *and* $\frac{P_n}{Q_n} = [c_0, c_1, \ldots, c_n]$ *(possibly unordered) isolates a positive real root of A, say* γ_i*. Then, in order for Th. 4 to hold, it suffices to consider, instead of* Δ*, the quantity* $|\gamma_i - \gamma_{c_i}|$*, where* γ_{c_i} *is the (complex) root of A closest to* γ_i*. When the transformed polynomial has no sign variation and* $[c_0, c_1, \ldots, c_n]$ *is the continued fraction expansion of the (positive) real part of a complex root of A, say* γ_i*, then again it suffices to replace* Δ *by* $|\gamma_i - \gamma_{c_i}|$*.*

Theorem 8 *The CF algorithm performs at most* $\mathcal{O}(d^2 + d\tau)$ *transformation steps.*

Proof. Let $0 < |\gamma_1| \leq |\gamma_2| \leq \cdots \leq |\gamma_k|$, $k \leq d$ be the (complex) roots of A with positive real part and let γ_{c_i} denote the root of A that is closest to γ_i.

We consider the binary tree T generated during the execution of the CF algorithm. The number of steps of the CF algorithm corresponds to the number of nodes in T, which we denote by $\#(T)$. We use some arguments and the notation from [19] in order to prune the tree.

With each node v of T we associate a Möbius transformation $M_v : X \mapsto \frac{kX+l}{mX+n}$, a polynomial A_v and implicitly an interval I_v whose unordered endpoints can be found if we evaluate M_v on 0 and on ∞. Recall that A_v is produced after M_v is applied to A. The root of T is associated with A, $M(X) = X$ (i.e $k = n = 1, l = m = 0$) and implicitly with the interval $(0, \infty)$.

Let a leaf u of T be of **type-i** if its interval I_u contains $i \geq 0$ real roots. Since the algorithm terminates the leaves are of type-0 or type-1. We will prune certain leaves of T so as to obtain a certain sub-tree T' where it is easy to count the number of nodes. We remove every leaf that has a sibling that is not a leaf. Now we consider the leaves that have a sibling that is also a leaf. If both leaves are of type-1, we arbitrary prune one of them. If one of them is of type-1 then we prune the other. If both leaves are of type-0, this means that the polynomial on the parent node has at least two sign variations and thus that we are trying to isolate the (positive) real part of some complex root. We keep the leaf that contains the (positive) real part of this root. And so $\#(T) < 2\,\#(T')$.

Now we consider the leaves of T'. All are of type-0 or type-1. In both cases they hold the positive real part of a root of A, the associated interval is $|I_v| \geq |\gamma_i - \gamma_{c_i}|$ (Rem. 7) and the number of nodes from a leaf to the root is n_i, which is such that the hypothesis of Th. 4 is satisfied. Since n_i is the smallest index such that the hypothesis of Th. 4 holds, if we reduce n_i by one then the inequality does not hold. Thus

$$F_{n_i-2}\, F_{n_i-1}\, |\gamma_i - \gamma_{c_i}| \leq \frac{2}{\sqrt{3}} \Rightarrow \phi^{2n_i-5}\, |\gamma_i - \gamma_{c_i}| < \frac{2}{\sqrt{3}} \Rightarrow n_i < 2 - \frac{1}{2} \lg |\gamma_i - \gamma_{c_i}|.$$

We sum over all n_i to bound the nodes of T', thus

$$\#(T') \leq \sum_{i=1}^{k} n_i \leq 2k - \frac{1}{2} \sum_{i=1}^{k} \lg |\gamma_i - \gamma_{c_i}| \leq 2k - \frac{1}{2} \lg \prod_{i=1}^{k} |\gamma_i - \gamma_{c_i}|. \tag{6}$$

In order to apply Th. 6 we should rearrange $\prod_{i=1}^{k} |\gamma_i - \gamma_{c_i}|$ so that the requirements on the indices of roots are fulfilled. This can not be achieved when symmetric products occur and thus the worst case is when the product consists only of symmetric products i.e $\prod_{i=1}^{k/2} |(\gamma_j - \gamma_{c_j})(\gamma_{c_j} - \gamma_j)|$. Thus we consider the square of the inequality of Th. 6 taking $\frac{k}{2}$ instead of k and $\mathtt{disc}(A) \geq 1$ (since A is square-free), thus

$$\begin{array}{rcl} \prod_{i=1}^{k} |\gamma_i - \gamma_{c_i}| & \geq & \left(2^{\frac{k}{2} - \frac{d(d-1)}{2}}\, \mathcal{M}(A)^{1-d-\frac{k}{2}}\right)^2 \\ -\lg \prod_{i=1}^{k} |\gamma_i - \gamma_{c_i}| & \leq & d^2 - d - k + (2d + k - 2) \lg \mathcal{M}(A) \end{array} \tag{7}$$

Eq. (6) becomes $\#(T') < 2k + d^2 - d - k + (2d + k - 2)\lg \mathcal{M}(A)$. However, for Mahler's measure it is known that $\mathcal{M}(A) \leq 2^\tau \sqrt{d+1} \Rightarrow \lg \mathcal{M}(A) \leq \tau + \lg d$, for $d \geq 2$, thus $\#(T') \leq 2k + d^2 - d - k + (2d + k - 2)(\tau + \lg d)$. Since $\#(T) < 2\,\#(T')$ and $k \leq d$, we conclude that $\#(T) = \mathcal{O}(d^2 + d\,\tau + d \lg d)$. □

4.1 Real root isolation

To complete the analysis of the CF algorithm we have to compute the cost of every step that the algorithm performs. In the worst case every step consists of a computation of a positive lower bound b (Line 10) and three transformations, $X \mapsto b + X$, $X \mapsto 1 + X$ and $X \mapsto \frac{1}{1+X}$ (Lines 11, 12 and 14 in Alg. 1). Recall, that inversion can be performed in $\mathcal{O}(d)$. Thus the complexity is dominated by the cost of the shift operation (Line 11 in Alg. 1) if a small number of calls to `PLB` is needed in order to compute a partial quotient. We will justify this in Sec. 4.2. In order to compute this cost a bound on $\mathcal{L}(c_k) \triangleq b_k, 0 \leq k \leq m_i$ is needed, see Eq. (2).

For the analysis of the CF algorithm we will need the following:

Theorem 9 (Fast Taylor shift) [48] *Let $A \in \mathbb{Z}[X]$, with $\deg(A) = d$ and $\mathcal{L}(A) = \tau$ and let $a \in \mathbb{Z}$, such that $\mathcal{L}(a) = \sigma$. Then the cost of computing $B = A(a + X) \in \mathbb{Z}[X]$ is $\mathcal{O}_B(\mathsf{M}\left(d^2 \lg d + d^2\sigma + d\tau\right))$. Moreover $\mathcal{L}(B) = \mathcal{O}(\tau + d\sigma)$.*

Initially A has degree d and bit size τ. Evidently the degree does not change after a shift operation. Each shift operation by a number of bit size b_h increases the bit size of the polynomial by an additive factor $d\,b_h$, in the worst case (Th. 9). At the $h-$th step of the algorithm the polynomial has bit size $\mathcal{O}(\tau + d\sum_{i=1}^{h} b_i)$ and we perform a shift operation by a number of bit size b_{h+1}. Th. 9 states that this can be done in $\mathcal{O}_B\left(\mathsf{M}\left(d^2 \lg d + d^2 b_{h+1} + d(\tau + d\sum_{i=1}^{h} b_i)\right)\right)$ or $\mathcal{O}_B\left(\mathsf{M}\left(d^2 \lg d + d\tau + d^2 \sum_{i=1}^{h+1} b_i\right)\right)$.

Now we have to bound $\sum_{i=1}^{h+1} b_i$. For this we use Eq. (2), which bounds $E[b_i]$. By linearity of expectation it follows that $E[\sum_{i=1}^{h+1} b_i] = \mathcal{O}(h)$ Since $h \leq \#(T) = \mathcal{O}(d^2 + d\tau)$ (Th. 8), the (expected) worst case cost of step h is $\mathcal{O}_B(\mathsf{M}\left(d^2 \lg d + d\tau + d^2(d^2 + d\tau)\right))$ or $\widetilde{\mathcal{O}}_B(d^2(d^2 + d\tau))$. Finally, multiplying by the number of steps, $\#(T)$, we conclude that the overall complexity is $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau + d^4\tau^2)$, or $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ if $d = \mathcal{O}(\tau)$, or $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$.

Now let us isolate, and compute the multiplicities, of the real roots of $A_{in} \in \mathbb{Z}[X]$, which is not necessarily square-free, with $\mathtt{deg}(A_{in}) = d$ and $\mathcal{L}(A_{in}) = \tau$. We use the technique from [21] and compute the square-free part A of A_{in} using Sturm-Habicht sequences in $\widetilde{\mathcal{O}}_B(d^2\tau)$. The bit size of A is $\mathcal{L}(A) = \mathcal{O}(d + \tau)$. Using the CF algorithm we isolate the positive real roots of A and then, by applying the map $X \mapsto -X$, we isolate the negative real roots. Finally, using the square-free factorization of A_{in}, which can be computed in $\widetilde{\mathcal{O}}_B(d^3\tau)$, it is possible to find the multiplicities in $\widetilde{\mathcal{O}}_B(d^3\tau)$.

The previous discussion leads to the following theorem.

Theorem 10 *Let $A \in \mathbb{Z}[X]$ (not necessarily square-free) such that* $\mathtt{deg}(A) = d > 2$ *and* $\mathcal{L}(A) = \tau$. *We can isolate the real roots of A and compute their multiplicities in expected time $\widetilde{\mathcal{O}}_B(d^6 + d^4\tau^2)$, or $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$.*

The same complexity bound can be achieved for non square-free polynomials if instead of using Sturm-Habicht sequences in order to compute to the square-free part of the polynomial and compute the multiplicities, we rely on Th. 5.

4.2 Rational roots and PLB (Positive Lower Bound) realization

This section studies a way to compute a lower bound on the positive roots and presents its efficiency and accuracy. It seems that this is the standard approach in CF algorithms, though it is seldom, if at all, discussed.

There are two issues that we have to discuss further.

The first one concerns the rational numbers. If the polynomial A has (only) rational real roots then their continued fraction expansion neither follows the Gauss-Kuzmin distribution nor Khintchine's law. However, recall that if $\frac{p}{q}$ is a root of A then p divides a_0 and q divides a_d, thus in the worst case $\mathcal{L}(p/q) = \mathcal{O}(\tau)$ and so the rational roots are isolated fast among themselves. Treating them as real algebraic numbers leads to an overestimation of the number of iterations.

The second issue concerns the number of calls of function PLB that must be applied in order to compute a partial quotient. We made the assumption that this number of calls is small. In practice this is always the case, except when the polynomial has only rational real roots, of great magnitude, well separated and we are interested in the practical complexity. In this case function PLB must be applied many times in order to compute a partial quotient. Richtmyer et al. [38] in order to overcome this situation perform a small number of Newton-like iterations in order to get a good approximation of the partial quotient. In [4], see also [2, 3], the problem was solved partially by applying the map $X \mapsto bX$, where b is the computed positive root bound, when $b \geq 16$. This is what we do in our implementation.

The assumption that the number of calls to PLB is small enough, is strengthened by (1), since it implies that the probability that a partial quotient is of magnitude ≤ 10 is ~ 0.87. This is why in practice the partial quotients are of very small magnitude. Moreover, the relation

$$\left|\gamma - \frac{P_n}{Q_n}\right| < \frac{1}{c_{n+1}\, Q_n^2},$$

implies that the appearance of a partial quotient of an extra-ordinary big magnitude means that the *previous* approximation of the algebraic number was extremely good.

However, the previous discussion does not provide a theoretical explanation. We will use results about the tightness of the positive root bounds in order to support our assumption.

Recall that a lower bound on the positive roots of a polynomial is computed as the inverse of the upper bound on the positive roots of the reciprocal polynomial. Thus in what follows we will consider only upper bounds for the positive roots. The bound that we will consider,

and that we also use in our implementation of PLB, is

$$B_2(A) := 2 \max_{a_k<0} \left\{ \left| \frac{a_k}{a_d} \right|^{\frac{1}{d-k}} \right\}, \tag{8}$$

where $0 \leq k < d$, which is due to [28], see also [26, 29]. Notice that B_2 is a bound for the positive roots only and not a bound for all the (complex) roots of the polynomial. For such bounds, the reader may refer to e.g [32, 33, 46]. For other bounds on the positive roots the reader may refer to [25, 28, 42].

If instead of B_2, we compute a bound B_1, taking into account all the coefficients then from [46] B_1 is at most d times the biggest root of A. Since B_2 is a smaller bound we can conclude that $\gamma \leq B_2 \leq B_1 \leq d \cdot \gamma$. A similar result can be obtained if we consider the bound of Hong [25] or any other bound that guarantees that it is $\mathcal{O}(d)$ times away from the biggest root, see e.g [23, 32, 33].

Last, but not least, we have to mention that the implementation of B_2 requires $\widetilde{\mathcal{O}}(d)$ arithmetic operations [5, 30, 46] and as van der Sluis [46] says for B_1 and thus for B_2, this bound "is to be recommended among all" because of its simplicity and the good quality of its results.

If b is the computed positive root bound and γ is the closest positive real root of A to it, which we are trying to isolate, then, from the previous discussion, it holds that $b \leq \gamma \leq d\, b$. Recall that the lower bound on the positive roots is obtained as an upper bound on the positive real roots of the reciprocal polynomial. One integer in the interval $[b, d\, b]$ is the partial quotient of γ that we actually want to compute. We can perform binary search based on Budan's theorem (Th. 3) in order to compute an interval $[c, c+1] \subseteq [b, d\, b]$ such that it contains γ and $c \in \mathbb{Z}$ is the partial quotient that we are intersted in. Budan's theorem corresponds to 2 polynomial shifts, thus the binary search needs, in order to compute c, at most $\mathcal{O}(\lg d + \lg b)$ shift operations. However, $b \leq c \Rightarrow \mathcal{L}(b) \leq \mathcal{L}(c) = \mathcal{O}(1)$, and since c is a partial quotient in the continued fraction expansion of γ, its magnitude should follow Khintchine, c.f (2).

The previous discussion implies that at every step of the algorithm we must perform, additionally $\mathcal{O}(\lg d)$ shift operations, instead of at most 2 that we assumed, in order to compute a partial quotient. Thus, the complexity of the algorithm should be multiplied by a factor $\lg d$, which does not change the bound $\widetilde{\mathcal{O}}_B(d^6 + d^4\tau^2)$.

In practice the tightness of the positive root bounds is usually very good.

4.3 Better complexity bounds

A closer look to the proof of Th. 8 reveals that in order to derive the number of steps of the CF algorithm we do not depend on an interval that initially contains all the real roots. Notice that this dependence is intrinsic for the subdivision algorithms [19, 21]. This will allow us to improve the complexity of the CF algorithm by spreading away the roots.

We consider the square-free polynomial A and we apply the homothetic transformation $X \mapsto X/2^{\ell(d+\tau)}$, where ℓ is a constant specified by (12). The transformed polynomial, say

C, has bit size $\mathcal{O}(\tau + \ell d^2 + \ell d\tau)$ and its roots β_j are the roots of A multiplied by $2^{\ell(d+\tau)}$, i.e

$$\beta_i = 2^{\ell(d+\tau)}\,\gamma_i, \tag{9}$$

where γ_i are the roots of A and $1 \leq i \leq d$. Evidently it suffices to isolate the real roots of C.

We will use the notation of the proof of Th. 8. Let k_1 be the number roots of C with positive real part and k_2 those with negative real part. Notice that $k_1 + k_2 = d$. Following the proof of Th. 8, see Eq. (6), the number of steps that the CF algorithm must perform in order to isolate the real roots of C is

$$\begin{aligned}
\#(T') &\leq \sum_{i=1}^{k_1} n_i + \sum_{j=1}^{k_2} n_j \\
&\leq 2k_1 - \frac{1}{2}\sum_{i=1}^{k_1} \lg|\beta_i - \beta_{c_i}| + 2k_2 - \frac{1}{2}\sum_{j=1}^{k_2} \lg|\beta_j - \beta_{c_j}| \\
&\leq 2d - \frac{1}{2}\lg\prod_{i=1}^{d}|\beta_i - \beta_{c_i}|.
\end{aligned} \tag{10}$$

If we consider the product term of the previous equation and (9) then

$$\prod_{i=1}^{d}|\beta_i - \beta_{c_i}| = 2^{\ell d^2 + \ell d\tau}\prod_{i=1}^{d}|\gamma_i - \gamma_{c_i}|.$$

Combining the previous equation with (7) we have

$$-\lg\prod_{i=1}^{d}|\beta_i - \beta_{c_i}| \leq d^2 + (3d-2)\tau - 2d - 2\lg d + 3d\lg d - \ell\left(d^2 + d\tau\right). \tag{11}$$

We want to specify the value of ℓ in such way so as to eliminate the quantities of the form d^2 and $d\tau$ from Eq. (11). By elementary calculus we see that ℓ should be

$$\ell = \frac{(3d-2)\tau + d^2 - 2d}{d\tau + d^2}. \tag{12}$$

If we make the assumption that $d = \mathcal{O}(\tau)$ then, using this result and combining Eq. (10) and (11), we conclude that $\#(T) = \mathcal{O}(d\,\lg d) = \widetilde{\mathcal{O}}_B(d)$. If we substitute this value of $\#(T)$ in the proof of Th. 10, presented in Sec. 4, and taking into account that the bit size of C is $\mathcal{O}(\tau + \ell d^2 + \ell d\tau)$ then we conclude that the expected complexity of this variant of the CF algorithm is $\widetilde{\mathcal{O}}_B(d^3\tau)$.

The previous variant of the CF algorithm has small practical interest because applying the homothetic transformation $X \mapsto X/2^{\ell(d+\tau)}$ to the polynomial increases its bit size a lot.

However, to the best of our knowledge this is the first complexity bound, even using average case analysis, that matches the complexity bounds of the numerical algorithms [36, 37, 41].

We conjecture that the expected complexity of the subdivision solvers, i.e Descartes/Bernstein and Sturm is also the same, but we will report on this in a future work.

5 Implementation and experiments

We have implemented the CF algorithm in SYNAPS [2] [35], which is a `C++` library for symbolic-numeric computations that provides data-structures, classes and operations for univariate and multivariate polynomials, vector and matrices. Our code will be included in the next major public release of SYNAPS. The implementation is based on the integer arithmetic of GMP[3] (v. 4.1.4) and uses only transformations of the form $X \mapsto 2^\beta X$ and $X \mapsto X + 1$ to benefit from the fast implementations that are available in GMP. However, our implementation follows the generic programming paradigm, thus any library that provides arbitrary precision integer arithmetic can be used instead of GMP.

We restrict ourselves to square-free polynomials of degree $\in \{100, 200, \ldots, 1000\}$. Following [40], the first class of experiments concerns well-known ill-conditioned polynomials namely: Laguerre (L), first (C1) and second (C2) kind Chebyshev, and Wilkinson (W) polynomials. We also consider Mignotte (M1) polynomials $X^d - 2(101X - 1)^2$, that have 4 real roots but two of them very close together, and a product $\left(X^d - 2(101X - 1)^2\right)\left(X^d - 2((101 + \frac{1}{101})X - 1)^2\right)$ of two such polynomials (M2) that has 8 real roots. Finally, we consider polynomials with random coefficients (R1), and monic polynomials with random coefficients (R2) in the range [-1000, 1000], produced by MAPLE, using 101 as a seed for the pseudo-random number generator.

We performed experiments against RS[4], which seems to be one of the fastest available software for exact real root isolation. It implements a subdivision-based algorithm using Descartes' rule of sign with several optimizations and symbolic-numeric techniques [40]. Note that we had to use RS through its MAPLE interface. Timings were reported by its internal function `rs_time()`.

We also test ABERTH [9, 10], which a numerical solver with unknown (bit) complexity but very efficient in practice, available through SYNAPS. In particular, it uses multi-precision floats and provides a floating-point approximation of all the complex roots. Since ABERTH is a numerical solver it approximates the roots up to a desired accuracy. Even though we tuned ABERTH to search for roots on the real axis only, unfortunately, we were not always able to tune its behavior in order to produce the correct number of real roots in all the cases, i.e to specify the output precision.

In SYNAPS, there are several univariate solvers, based on Sturm sequences, Descartes' rule of sign, Bernstein basis, etc (see [21] for details and experimental results). CF is clearly faster than all these solvers, therefore we do not report on these experiments. In particular,

[2] `www-sop.inria.fr/galaad/logiciels/synaps/`

[3] `www.swox.com/gmp/`

[4] `fgbrs.lip6.fr/salsa/Software/index.php`

		100	200	300	400	500	600	700	800	900	1000
L	CF	0.27	2.24	9.14	25.27	55.86	110.13	214.99	407.09	774.22	1376.34
	RS	0.65	3.65	13.06	35.23	77.21	151.17	283.43	527.42	885.86	1387.45
	#roots	100	200	300	400	500	600	700	800	900	1000
C1	CF	0.11	0.85	3.16	8.61	19.67	38.23	77.75	139.18	247.11	414.51
	RS	0.21	1.36	3.80	10.02	23.15	46.02	82.01	150.01	269.35	458.67
	#roots	100	200	300	400	500	600	700	800	900	1000
C2	CF	0.11	0.77	3.14	8.20	19.28	38.58	73.59	133.52	233.48	386.61
	RS	0.23	1.48	3.80	9.84	23.28	46.34	83.58	146.04	273.00	452.77
	#roots	100	200	300	400	500	600	700	800	900	1000
W	CF	0.11	0.76	2.54	6.09	12.07	21.43	34.52	53.35	81.88	120.21
	RS	0.09	0.59	2.25	6.34	14.62	29.82	55.47	104.56	179.23	298.45
	#roots	100	200	300	400	500	600	700	800	900	1000
M1	CF	0.02	0.08	0.21	0.42	0.73	1.19	1.84	2.75	4.16	6.22
	RS	7.83	287.27	1936.48	7328.86	*	*	*	*	*	*
	ABERTH	0.01	0.04	0.07	0.11	0.12	0.26	0.43	0.37	0.47	0.90
	#roots	4	4	4	4	4	4	4	4	4	4
M2	CF	0.08	0.43	1.10	2.78	4.71	8.67	18.26	25.28	40.15	60.10
	RS	1.24	144.64	1036.785	4278.275	12743.79	*	*	*	*	*
	#roots	8	8	8	8	8	8	8	8	8	8
R1	CF	0.001	0.04	0.07	0.33	0.06	0.37	0.66	0.76	1.03	1.77
	RS	0.026	0.09	0.11	0.68	0.22	0.89	0.95	0.69	1.55	2.09
	ABERTH	0.02	0.03	0.07	0.14	0.21	0.31	0.44	0.51	0.64	0.80
	#roots	4	4	2	6	2	4	4	2	4	4
R2	CF	0.01	0.04	0.08	0.36	0.14	0.38	0.74	0.77	1.24	1.42
	RS	0.05	0.23	0.47	1.18	0.81	1.64	2.68	3.02	4.02	4.88
	ABERTH	0.01	0.05	0.08	0.14	0.23	0.33	0.44	0.55	0.67	0.83
	#roots	4	4	4	6	4	4	6	4	6	4

Table 1: Experimental results

the large inputs used here are not tractable by the Sturm-sequence solver in SYNAPS, and this is also the case for another implementation of the Sturm-sequence solver in CORE[5].

So, in Table 1, we report experiments with CF, RS, ABERTH, where the timings are in seconds. The asterisk (*) denotes that the computation did not finish after 12000s. The experiments were performed on a 2.6 GHz Pentium with 1 GB RAM, and our code was compiled using g++ 3.3 with options -O3 -DNDEBUG.

For (M1) and (M2), there are rational numbers with a very simple continued fraction expansion that isolate the real roots which are close. These experiments are extremely hard for RS. On (M1), ABERTH is the fastest and correctly computes all real roots, but on (M2), which has 4 real roots close together, it is slower than CF. CF is advantageous on (W) since, as soon as a real root is found, transformations of the form $X \mapsto X + 1$ rapidly produce the other real roots. We were not able to tune ABERTH on (W). For (L), (C1) and (C2), CF is clearly faster than RS, while we were not able to appropriately tune ABERTH to produce the correct number of real roots. The polynomials in (R1) and (R2) have few and well separated real roots, thus the semi-numerical techniques in RS are very effective. To be more specific, RS isolates all roots using only 63 bits of accuracy (this information was extracted using the function `rs_verbose( 1)`). However, even in this case, CF is comparable

[5] `cs.nyu.edu/exact/core_pages/`

to RS. ABERTH is even faster on these experiments (see Table 1). We have to mention as F. Rouillier pointed out to us that RS can about 30% faster in (L), (C1) and (C2) if we use it with the (non-default) option `precision=0`.

We finally tested a univariate polynomial that appears in the Voronoi diagram of ellipses [22]. The polynomial has degree 184, coefficient bit size 903, and 8 real roots. CF solves it in 0.12s, RS in 0.3s and ABERTH in 1.7s.

In short, CF is complete, simple to use and is at least as efficient as the state of the art. There are ways to improve our solver. First, instead of exact integer arithmetic we may use semi-numerical techniques like those in RS [40]. These techniques may be based on interval arithmetic.

Acknowledgments Both authors acknowledge fruitful discussions with Alkiviadis Akritas and Bernard Mourrain. The first author is grateful to Maurice Mignotte for discussions about the separation bound, to Doru Stefanescu for various discussions and suggestions about the bounds of the positive roots of polynomials and to Fabrice Rouillier for various discussions about RS and the experiments. Both authors acknowledge partial support by IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes) and by a doctoral research grant by the Greek General Secretariat for Research and Technology under the program $\Pi ENE\Delta$ 03.

References

[1] A. Akritas. There is no "Uspensky's method". Extended Abstract. In *Proc. Symposium on Symbolic and Algebraic Computation*, pages 88–90, Waterloo, Ontario, Canada, 1986.

[2] A. Akritas. An implementation of Vincent's theorem. *Numerische Mathematik*, 36: 53–62, 1980.

[3] A. Akritas and A. Strzebonski. A comparative study of two real root isolation methods. *Nonlinear Analysis: Modelling and Control*, 10(4):297–304, 2005.

[4] A. Akritas, A. Bocharov, and A. Strzébonski. Implementation of real root isolation algorithms in Mathematica. In *Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (Interval '94)*, pages 23–27, St. Petersburg, Russia, March 1994.

[5] A.G. Akritas. *Elements of Computer Algebra with Applications*. J. Wiley & Sons, New York, 1989.

[6] A. Alesina and M. Galuzzi. A new proof of Vincent's theorem. *L'Enseignement Mathématique*, 44:219–256, 1998.

[7] D. Bailey, J. Borwein, and R. Crandall. On the Khintchine Constant. *Mathematics of Computation*, 66:417–431, 1997.

[8] S. Basu, R. Pollack, and M-F.Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2003. ISBN 3-540-00973-6.

[9] D. Bini. Numerical computation of polynomial zeros by means of Aberth's method. *Numerical Algorithms*, 13(3–4):179–200, 1996.

[10] D. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, pages 127–173, 2000.

[11] E. Bombieri and A. van der Poorten. Continued fractions of algebraic numbers. In *Computational algebra and number theory (Sydney, 1992)*, pages 137–152. Kluwer Acad. Publ., Dordrecht, 1995.

[12] R. Brent, A. van der Poorten, and H. Riele. A Comparative Study of Algorithms for Computing Continued Fractions of Algebraic Numbers. In Henri Cohen, editor, *ANTS*, LNCS, pages 35–47. Springer, 1996.

[13] D. Cantor, P. Galyean, and H. Zimmer. A continued fraction algorithm for real algebraic numbers. *Mathematics of Computation*, 26(119):785–791, July 1972. ISSN 0025-5718.

[14] J. Chen. A new algorithm for the isolation of real roots of polynomial equations. In *Proc. 2nd International Conference on Computers and Applications*, pages 714–719. IEEE Computer Soc. press, 1987.

[15] G. Collins and A. Akritas. Polynomial real root isolation using Descartes' rule of signs. In *SYMSAC '76*, pages 272–275, New York, USA, 1976. ACM Press.

[16] G.E. Collins and R. Loos. Real zeros of polynomials. In B. Buchberger, G.E. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 83–94. Springer-Verlag, Wien, 2nd edition, 1982.

[17] J. H. Davenport. Cylindrical algebraic decomposition. Technical Report 88–10, School of Mathematical Sciences, University of Bath, England, available at: http://www.bath.ac.uk/masjhd/, 1988.

[18] Z. Du, V. Sharma, and C. K. Yap. Amortized bound for root isolation via Sturm sequences. In D. Wang and L. Zhi, editors, *Int. Workshop on Symbolic Numeric Computing*, pages 81–93, School of Science, Beihang University, Beijing, China, 2005.

[19] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the descartes method. In *ISSAC '06: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pages 71–78, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-276-3.

[20] I. Z. Emiris and E. P. Tsigaridas. Computing with real algebraic numbers of small degree. In S. Albers and T. Radzik, editors, *Proc. ESA*, volume 3221 of *LNCS*, pages 652–663. Springer Verlag, 2004.

[21] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real Algebraic Numbers: Complexity Analysis and Experimentation. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, editors, *Reliable Implementations of Real Number Algorithms: Theory and Practice*, LNCS (to appear). Springer Verlag, 2006. also available in www.inria.fr/rrrt/rr-5897.html.

[22] I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. The predicates for the Voronoi diagram of ellipses. In *Proc. 22th Annual ACM Symp. on Computational Geometry*, pages 227–236, Sedona, USA, 2006.

[23] P. Henrici. *Applied and computational complex analysis*. John Wiley & Sons, New York, NY, 1977.

[24] D. Hensley. The largest digit in the continued fraction expansion of a rational number. *Pacific Journal of Mathematics*, 151(2):237–255, 1991.

[25] H. Hong. Bounds for absolute positiveness of multivariate polynomials. *Journal of Symbolic Computation*, 25(5):571–585, May 1998.

[26] J. R. Johnson. *Algorithms for Polynomial Real Root Isolation*. PhD thesis, The Ohio State University, 1991.

[27] A. Khintchine. *Continued Fractions*. University of Chicago Press, Chicago, 1964.

[28] J. Kioustelidis. Bounds for the positive roots of polynomials. *Journal of Computational and Applied Mathematics*, 16:241–244, 1986.

[29] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 3rd edition, 1998.

[30] W. Krandick. Isolierung reeller nullstellen von polynomen. In J. Herzberger, editor, *Wissenschaftliches Rechnen*, pages 105–154. Akademie-Verlag, Berlin, 1995.

[31] W. Krandick and K. Mehlhorn. New bounds for the Descartes method. *JSC*, 41(1): 49–66, Jan 2006.

[32] M. Mignotte. *Mathematics for computer algebra*. Springer-Verlag, New York, 1991.

[33] M. Mignotte and D. Stefanescu. *Polynomials: An algorithmic approach*. Springer, 1999.

[34] Maurice Mignotte. On the Distance Between the Roots of a Polynomial. *Appl. Algebra Eng. Commun. Comput.*, 6(6):327–332, 1995.

[35] B. Mourrain, J. P. Pavone, P. Trébuchet, and E. Tsigaridas. SYNAPS, a library for symbolic-numeric computation. In *8th Int. Symposium on Effective Methods in Algebraic Geometry, MEGA*, Sardinia, Italy, May 2005. Software presentation.

[36] V.Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and rootfinding. *J. Symbolic Computation*, 33(5):701–733, 2002.

[37] V.Y. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM Rev.*, 39(2):187–220, 1997.

[38] R. Richtmyer, M. Devaney, and N. Metropolis. Continued fraction expansions of algebraic numbers. *Mumerische Mathematik*, 4:68–64, 1962.

[39] D. Rosen and J. Shallit. A continued fraction algorithm for approximating all real polynomial roots. *Math. Mag*, 51:112–116, 1978.

[40] F. Rouillier and Z. Zimmermann. Efficient isolation of polynomial's real roots. *J. of Computational and Applied Mathematics*, 162(1):33–50, 2004.

[41] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany, 1982.

[42] D. Stefanescu. New bounds for the positive roots of polynomials. *Journal of Universal Computer Science*, 11(12):2132–2141, 2005.

[43] E. P. Tsigaridas and I. Z. Emiris. Univariate polynomial real root isolation: Continued fractions revisited. In Y. Azar and T. Erlebach, editors, *In Proc. 14th European Symposium of Algorithms (ESA)*, volume 4168 of *LNCS*, pages 817–828, Zurich, Switzerland, 2006. Springer Verlag.

[44] J. V. Uspensky. *Theory of Equations*. McGraw-Hill, 1948.

[45] A. van der Poorten. An introduction to continued fractions. In *Diophantine analysis*, pages 99–138. Cambridge University Press, 1986.

[46] A. van der Sluis. Upper bounds for the roots of polynomials. *Numerische Mathematik*, 15:250–262, 1970.

[47] A. J. H. Vincent. Sur la résolution des équations numériques. *J. Math. Pures Appl.*, 1: 341–372, 1836.

[48] J. von zur Gathen and J. Gerhard. Fast Algorithms for Taylor Shifts and Certain Difference Equations. In *ISSAC*, pages 40–47, 1997.

[49] X. Wang. A method for isolating roots of algebraic equations. Number N. 1. 1960. Univ. Academic Press.

[50] C.K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399